REMARKS

Amendments.

The applicant thanks the examiner for the suggestion in numbered paragraph 14 of the office action, and has amended various claims accordingly, by specifying the context to be a transaction context.

By another amendment, the limitation regarding concurrency preferences disclosed in claim 18 has been included in claim 7.

The following amendments are ordered according to the objections and rejections of the office action, pages 2-4.

- Claim 18, line 5: delete the superfluous "or".
- Claim 18, line 6: use "its or his".
- Claim 23: insert the "and" as suggested by the examiner.
- Claim 32: use "logged by storing on persistent storage".
- Claim 37: delete the "in the end" after "globalCommit"
- Claim 7: use "a root transaction"
- Claim 8: change the beginning of the claim to "A method for use with a data management system, …", bringing it in line with claim 10, which complements claim 8 by differing in the alternative expressed in the last line.
- Claim 18: please change the three phrases mentioned by the examiner to use the indefinite article.
- Claim 24: please use "an order of their original counterparts".
- Claim 25: delete the "the" before "corresponding".
- Claim 28: delete the "the" before "data managed".

The following amendments are ordered according to the claim numbers and serve to more clearly distinguish the claimed subject matter from the prior art.

- Claim 7:
    - line 2: following "client needs", insert "wherein the concurrency preferences specify the extent to which shared resource access is desired or allowed or denied among descendant transaction invocations of the root invocation or user and other, concurrent transaction invocations who are also descendants of the same root"
- Claim 8:
    - line 5: insert "transactional" before "service"

- line 6: insert “transactional” before “service”
- line 7: insert “transactional” before “service”
- line 12: insert “transactional” before “service”
- line 19: insert “transactional” before “invocations”
- line 22: insert “transactional” before “invocations”

- Claim 12:
  - line 1, after “A distributed system” add “using a two-phase commit protocol”.
  - line 5, replace “globalCommit message” by “two-phase commit message”
- Claim 18:
  - line 3: insert “transactional” before “service”
  - line 5: insert “transactional” before “invocation”
  - line 6: insert “transactional” before “invocation”
  - line 11: insert “transactional” **after** “descendant”
  - line 11: insert “transactional” **after** “concurrent”
- Claim 23:
  - line 1: insert “transactional” before “service”
  - line 3. delete “contexts or”, as suggested by the examiner
  - line 4. insert “of the service”, before “, by a remote client”
  - line 5: insert “transaction” before “context or contexts”
  - line 6. insert “of the service”, before “, by a remote client”
  - line 8: insert “transaction” before “context”
  - line 10: insert “transactional” before “service”
  - line 10: insert “locally” before “maintaining”
  - line 11: insert “transaction” before “context”
  - line 11: insert “locally maintained” before “undo”
  - line 12: insert “transactional” before “service”
- Claim 26:
  - lines 1-2: replace “to a remote client or its context” by “to the transactional context of a remote client””
- Claim 27:
  - line 2: insert “transaction” before “context”
- Claim 28:
  - line 2: insert “transactional” before “service”
- Claim 29:

- line 1: insert “on transactions” before “to ensure”

Formalities

It is thus suggested that the amended claims overcome the objections and §112 rejections.

Rejection of claims 6-11.

Rich was filed January 5, 1999 and was first published September 24, 2002 (upon issue).

Fouquet was filed July 10, 1997 and was first published August 7, 2001 (upon issue).

Applicant's filing date is at least as early as November 2, 2001 and the effective filing date may be as early as November 2, 2000.

At paragraph 11, the Examiner rejects claims 6-11 on the view that as of applicant's filing date, it would supposedly have been obvious to one skilled in the art to combine the teachings of Rich and Fouquet. The problem with this argument by the Examiner is that as of applicant's filing date, Rich was secret within the USPTO. It would first have been available to those skilled in the art only on September 24, 2002.

What's more, applicant's effective filing date may well be November 2, 2000, in which case as of applicant's effective filing date, Fouquet would likewise have been unavailable to those skilled in the art.

So in fact as of applicant's filing date it would not have been possible for one skilled in the art to combine the two references.

By reason of the foregoing the rejection of claims 6-11 over a two-way combination of Rich and Fouquet should be withdrawn.

Reconsideration is requested.

§102 rejections

Claim 12

Claim 12 has been amended to distinguish it from the prior art, specifying that a two-phase commit protocol is used. This mechanism is explained in the application, beginning at paragraph 0014, and details with relation to this mechanism are discussed throughout the specification.

Since Camp explicitly rejects the use of a two-phase commit protocol (col. 3, lines 55-60 and col. 4, lines 20-25), the subject matter of claim 12, which is directed to a two-phase commit protocol, and recites a two-phase commit message carrying information about the actual work being committed is not disclosed or suggested by Camp.

Claim 18

Applicant respectfully disagrees that Rich discloses setting and propagating concurrency preferences: As claimed, the concurrency preferences

> *specify the extent to which shared resource access is desired or allowed or denied among descendant transaction invocations of the root invocation or user and other, concurrent transaction invocations who are also descendants of the same root.*

This means that the concurrency preferences control the invocation of subtransactions and therefore the control the transaction process during the execution of the transaction itself.

In contrast, the in the cited section of Rich (col. 6, line 66 – col. 8, line 18) refers to a mechanism for propagating changes that are made (by a transaction) to a copied version of an object to a persistent version of the object. Therefore, Rich addresses the propagation of values stored in an object, whereas the present invention propagates preferences that affect how the transactions(s) are executed. Rich propagates any kind of data, regardless of its meaning, and does not care what the data stands for, whereas the present invention propagates concurrency preferences and uses them to control access to resources used by transactions that are descendants of the same root transaction.

Therefore, Rich neither discloses nor suggests the subject matter of claim 18.

§103 rejections

Claim 6

Applicant respectfully disagrees that Rich discloses

> *each process further characterized in that each transaction local thereto is independently handled at the process;*

since Rich, in the text quoted by the examiner (col 10, lines 31-41), discloses exactly the opposite: the common objects are replicated to separate workstations from a departmental server, and the integrity and consistency of these replicated objects is automatically maintained. It follows logically that for maintaing the consistency of separate objects, there must be some communication between the processes handling the objects, and therefore it cannot be said that the transactions are handled independently – they rather are dependent on each other and on the central server.

Applicant respectfully disagrees that Rich discloses

> *each process making scheduling and recovery decisions independent of any centralized component*

since Rich, in the text quoted by the examiner, only discloses a "shared transaction" which exists as various read-only copies and can be viewed (but not modified) by separate application programs. Nothing at all is said about each process making scheduling and recovery decisions.

The examiner argues that Fouquet discloses that conflicting transactions are not executed concurrently. The applicant disagrees: the referenced Fouquet work simply states that *operations* (i.e. parts) of conflicting transactions are not executed concurrently (col. 3, lines 1-2), but the transactions themselves are concurrent since one is started before the other terminates.

Claim 7

Claim 7 has been amended to define the term “concurrency preferences”. Consequently, the above reasoning with regard to claim 18 applies here as well: Rich neither discloses nor suggests concurrency preferences as claimed in claim 7.

Claims 8-11

Claim 8 has been amended to make the distinction from the prior art more explicit. The examiner argues that Fouquet anticipates "propagating ... a number or counter of invocations (i.e. subtransactions) ... on behalf of the root transaction". The applicant disagrees: Fouquet propagates a counter of conflicting *operations* of *other* transactions (not the same root) - see abstract lines 1-10.

In other words, Fouquet does not disclose nested transactions, that is, root transactions giving rise to child transactions or subtransactions. When describing the Fouquet system in the terms of the present application, then each of the Fouquet transactions is a separate root transaction having no subtransactions Therefore, Fouquet on the one hand does not have any subtransactions to count, and on the other hand states that it is *operations* (which are non-transactional) of transactions are counted.

Furthermore, this number is used for other purposes than in the present application: in Fouquet, it serves to activate (schedule) conflicting operations (abstract, last lines). In the present invention, said number/counter serves to decide whether or not global commit of the root and all its subtransactions is allowed (or aborted). This happens after all the operations have been scheduled. In Fouquet however, the number serves to decide if and when just one particular operation (which in turn is part of a transaction) may be activated. This is clearly different from a commit operation affecting an entire tree of transactions.

This difference is effective for claims 9-11 as well: In each case, the counter according to the invention, and the use of the counter for committing or aborting the global transaction is not disclosed by the Fouquet system, which only uses the counter for controlling an operation which is part of a transaction.

Claims 23-24

The basic differences to Gupta have already been shown in a prior response, dated July 18, 2006. Please refer to this response, section 3, for the explanation of how the Gupta system operates.

To come even close to the present invention, Gupta would have to disclose distributed transaction management *between transaction servers*, and nested SAGAs (using Gupta's terminology) with local UNDO at each transaction server, none of which is the case.

Gupta's invention is about distributed transactions managed *within one single* transaction server (col 3, liness 1-5, col 13, lines 25-30) Gupta does not disclose how to do what the present invention does: invoking a service with a transaction context – as in the present invention - wherein the context comes *from another transaction server*! Gupta does not disclose this, nor are his contexts nested (see Figure 5). Therefore, each of the features of claim 23 related to remote clients and invocations, such as

- *One or more operations that can be invoked by **remote** clients;*
- *Some or all such **remote** clients having one or more associated contexts or transaction contexts;*
- *An invocation by a **remote** client also containing partial or complete information indicating or containing said client"s context or contexts;*
- *An invocation, by a **remote** client, of an operation leading to a new transaction different from, but possibly related to, any existing client transaction;*

is not disclosed by Gupta.

Gupta does have "subtransactions" but these are API-level, local subtransactions that do not recognize the global transaction context (APIs are "transaction-agnostic", that is, they do not have a transactional interface; see col 6, lines 53-56 and again on lines 57-60). Gupta's subtransactions are not spread over several transaction servers; instead, Gupta's root transaction is centralized and the subtransactions do not recognize the global transaction context (transaction-agnostic API).

Gupta's undo is also not "local" since it is one dedicated central server that manages the undo information of all distributed applications.

In the present invention, subtransactions at some transaction server can be undone *independently of the root transaction,* since they are maintained locally by the service handling the substransaction. Claim 23 has been amended to reflect this explicitly. Please note that the service as described in claim 23 is invoked by a remote client, and therefore is itself a subtransaction. This is also not disclosed in Gupta, since the root transaction server maintains all undo information and executes all undoes in reverse order if the root fails (note: in Gupta, all transactions of the transaction server are root transactions). Also see Figure 4a .

For an example that illustrates the decentralized nature of the inventive system, and in particular the local storage of undo information, please refer to the attached figure and the following list of steps as executed according to the invention:

1. A transaction is started in Transaction Service 1: T1
2. Any local updates are done in the database: L1
3. Undo information is kept by the local transaction service
4. Another transactional service is invoked with context T1
5. Transactional Service 2 recognizes context T1 and creates subtransaction T2
6. Transactional Service 2 creates local subtransaction T2
7. Transaction Service 2 performs local updates: L2
8. Transaction Service 2 maintains its own undo data for L2
9. At this time, multiple things can happen:
    A. T2 can rollback (undo) without affecting T1. Not all undo transactions are done, only those of T2. Furthermore, T1 has no knowledge of the undo information for T2.
    B. T2 can wait for globalCommit (2-phase commit) to be performed by T1, it's parent transaction. T1 can decide to rollback (undo) its local work, and ask T2 to do the same with its own (local) work. Alternatively, T1 may decide to commit and ask T2 to do the same.

The example shows how, according to the invention, transaction contexts and transaction servers inter-operate. Services are transactional, and understand transactional calls (contexts). There is no centralized transaction service like in Gupta.

There is a totally different notion of client in Gupta: the present invention describes a building block (for a transaction service) that can be combined into peer-to-peer transactional service networks where every node manages its own transactions. A node becomes a client if it calls another node, and it remains a transaction server for its own work. This is not in Gupta. Gupta puts all the transactions into one node, has non-transactional clients, and does not ship a transaction context from/to clients.

In summary, the Gupta system aims at avoiding the need for a transaction context propagation by using a centralized server to manage transactions and undo. As such, it contradicts the present

invention.

Also, for this reason combining Rich with Gupta does not make sense: even if you would add a transaction context capability to the APIs in Gupta then you would still have the centrally managed undo operations. Making undo a distributed, nested and independently handled set of events is not trivial since Gupta does not distribute nor nest undo, nor does Rich talk about it. Yet this is exactly what the present invention creates when you combine its transactional services in a network: It creates a peer-to-peer distributed transaction network with nested and local undo.

In consequence, Gupta does not show both the remote client and the local service being transactional, and therefore Gupta discloses none of the features of claim 23. Furthermore, Gupta does provide a basis to which Rich may be applied, and the combination of Gupta and Rich does not lead to the subject matter of claim 23.

Claim 25

The applicant disagrees: Gupta does NOT allow the undo operations to be in the same system: the undo is managed in the transaction server (see col 6, lines 12-16), whereas the original operations are executed in the non-transactional API client.

Claim 26

The applicant disagrees: The term "client" in the present invention is an independent networked entity that calls a transactional service. Clients may be transaction-aware just like the servers. In Gupta, there is only one transaction service, and it is NOT the client. In Gupta, a client not knowing undo is trivial (by definition). The present invention's clients are transactional, so not knowing how to undo is NOT trivial.

Claim 27

The applicant disagrees: Since Gupta has no notion of transactional clients this claim cannot be anticipated.

Claim 28

The applicant disagrees: By "involved other services" applicant means "involved other *transactional* services". The claim has been amended accordingly. Gupta only recognizes (and strives for) one transactional service. Consequently, he cannot anticipate this claim.

Claim 29

The applicant disagrees: First of all, Gupta does not show claim 23. Second, Gupta's locks are only of the duration of the SUBtransaction (col 9, line 30). The present invention's locks are capable to hold afterwards, since undo only makes sense AFTER the subtransaction completes. This is clearly different. Since Gupta also states that undo is done only after subtransaction commit, Gupta's locks – which are forgotten after the subtransaction is committed, cannot serve the purpose of guaranteeing undoability after commitment.

Reconsideration is requested.

Respectfully submitted,

/s/

Carl Oppedahl
PTO Reg. No. 32746
telephone 970 468 8600